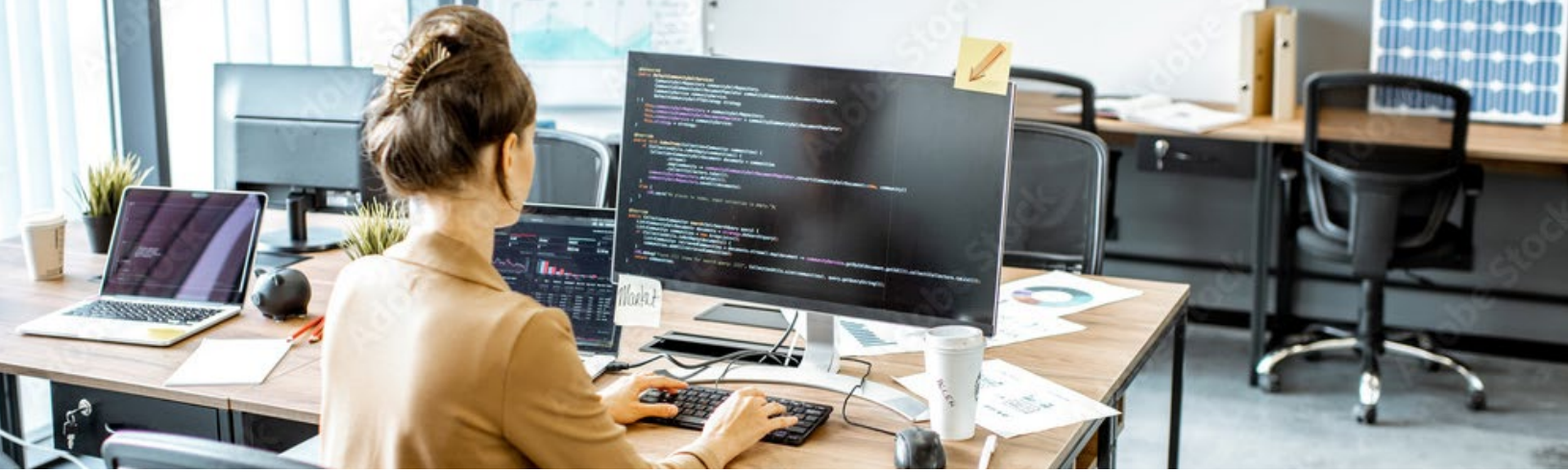
AWS FOR AUTOMOTIVE

# Accelerating software-defined vehicles through cloud-to-vehicle edge environmental parity

Girish Shirasat
Director of Software Strategy and Architecture, Automotive, Arm

Stefano Marzani
Principal Specialist Solutions Architect, Autonomous Vehicles, AWS

in partnership with



## Vision

It's 2025. Judith, a software engineer at a major automotive supplier, is working from home to address feedback from customers on an automatic cruise control function. She works in a cloud workspace, accessing a huge set of data related to the problematic behavior and utilizing this to adapt the control algorithm. She uses a software-in-the-loop (SiL) test bench with a vehicle dynamics simulator to evaluate and validate the new performance, then marks her package ready to deploy. At the same time, Mark—thousands of miles away on a different continent working for a design company—is making final touches to UI software being localized in a new region where a vehicle is going to be launched next week. He makes the modifications in a cloud panel provided by the OEM, running a massively parallel test suite leveraging virtual agents. On passing this, he marks the new UI package as ready for deployment. Kay, a development operations (DevOps) validation engineer, is working at the OEM's headquarters. He sees the two new software packages and runs the final hardware-in-the-loop (HiL) test suite to fully validate and certify the content to be deployed to the entire production vehicle fleet in the upcoming weekly release.

## Introduction

As the automotive industry embarks on a software-defined future, a vision pursued by many OEMs is to be able to develop software with the agility and flexibility described in the above narrative, delivering functionality incrementally into the vehicle with no compromise on quality or safety.

A cloud-native approach [1], able to preserve automotive-specific characteristics in terms of functional safety concepts and real-time execution, is key to creating such a software-centric ecosystem with modern digital services and in-vehicle user-friendly applications. With innovative and efficient workflows, it's an approach that enables more developers to be involved in the development process. It also enables automotive companies to shorten development time and achieve the agility needed to rapidly evolve and update features to meet the pace of modern consumer expectations.

A first key technical enabler to implementing automotive cloud-native development pipelines is achieving environmental parity between cloud environments and the target-embedded automotive edge platforms for eventually deploying the workload. As described by leading information architect Kevin Hoffman [2]:

***"The purpose of applying rigor and discipline to environmental parity is to give your team and your entire organization the confidence that the application will work everywhere."***

Achieving this parity allows development, verification, and validation in the cloud, independent of embedded hardware on developers' desks, and radically reduces the time to market for solutions across the automotive value chain, making applications portable and the development workflow future-proof.

To achieve environmental parity, and to support the increased demand for software in vehicles [3], a multi-core well-dimensioned embedded system is required inside the vehicles. In this paper, we look at how to implement a cloud-native approach to automotive system development, focusing particularly on achieving environmental parity between the execution environments in the cloud and at the vehicle edge, and we explore the impact of this approach in accelerating the time to market of software-defined vehicle trends [4].

## What is a software-defined vehicle (SDV)?

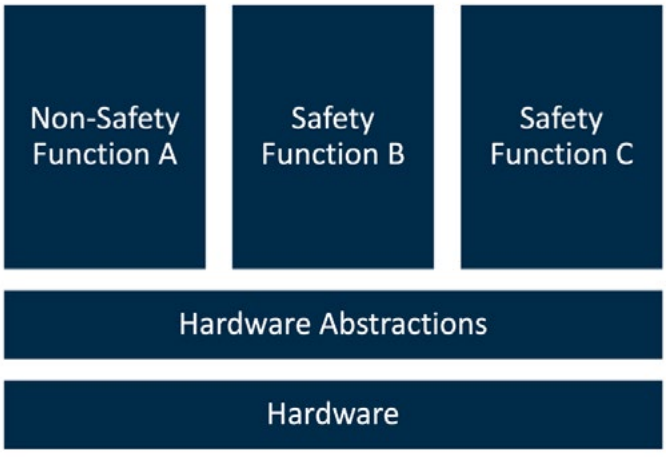



Figure 1 — Vehicle functions on top of abstracted hardware

"Software-defined" as it relates to a software-defined vehicle can be described as the characterization and implementation of vehicle features as software functions and services running on shared or centralized compute, as opposed to being implemented as individual physical electronic control units (ECUs) or similar. Furthermore, "software-defined" also infers the capability to develop and deploy these software-defined functions in an agile way during the entire automotive system development lifecycle, including pre- and post-manufacture. As shown in Figure 1, these software services in an ideal world would be hardware- and vendor-agnostsic and interpreted as a set of data-providing (sensors), data-processing (application logic), and data-consuming (actuators) services enabling specific functionality. This level of flexibility allows not just the OEMs but also pure-play software companies to create new functionality for a specific application domain. The ability to add new functionality post-sale has been a widespread practice in segments like smartphones for at least a decade, and there is now a huge appetite to enable a similar approach for traditional embedded markets, including automotive.

SDV could be summarized as systems that comprise of the following characteristics:

- A vehicle provided with connectivity features that is able to continuously transmit data and receive software updates over the air in a "big loop" perspective [5].
- Software abstracted from underlying hardware.
- Vehicle functions and capabilities enabled through software, upgradable and manageable throughout the vehicle lifecycle.
- Adoption of cloud-native design paradigms across the hardware platforms from the cloud to the vehicle edge, in a DevOps perspective.
- Mixed-critical management of automotive applications, enabling workloads with different levels of assurance against failure [6].

## Cloud-native in SDV

"Software-defined" isn't a new concept. In addition to the smartphone example above, it has been existent in telecommunications and data centers through software-defined networking, software-defined storage, and software-defined compute design paradigms, along with being prevalent in the enterprise application domain. In these domains, cloud-native concepts have been successfully used to deliver software-defined systems through a wealth of design patterns and ecosystem tools [7]. It is thus quite logical to explore what it means to apply cloud-native concepts in the automotive space such that the vast technical and commercial ecosystem developed around it can be leveraged to increase the effectiveness and speed of innovation.

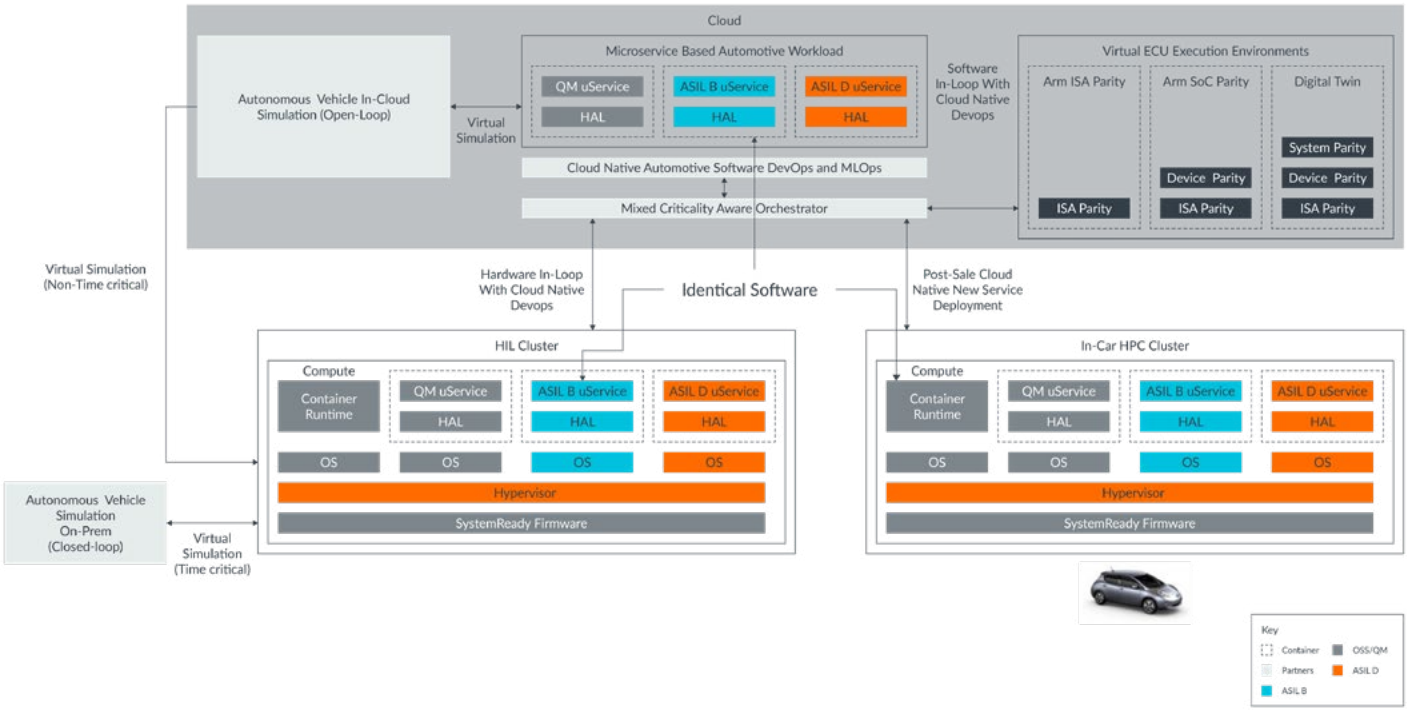



Figure 2 — Cloud-native applied to automotive system development

Figure 2 makes it easy to imagine a developer sitting in a café with their laptop, connecting to the cloud, developing their infotainment or advanced driving assistance system (ADAS), such as adaptive cruise control or lateral line control, application, committing the code, and triggering a build and integration cycle in the cloud on a virtual execution environment closely representing the physical automotive target system, assuming environmental parity between the cloud and the automotive edge. The level of environmental parity that can be achieved between the cloud and the automotive edge and the speed of the simulation directly impact the developer's efficiency.

With automotive systems software complexity now beginning to exceed that found in the Boeing 787 Dreamliner [8], the automotive developer would likely prefer to use proven software design patterns that are service-oriented architecture/ microservice-based, where containers form the fundamental technology to achieve workload mobility and complexity [9,10]. Automotive application development presents unique challenges that have not historically been considered in enterprise or smartphone segments where utilization of containers is a norm. Automotive workloads can be mixed critical in nature where—depending upon safety and real-time requirements—the microservices forming the workload will be constrained by specific spatial and temporal requirements in addition to corresponding safety decomposition. Depending on the application, some of these microservices may demand requirements, such as being quality managed (QM) or achieving ASIL-B/ Avcv-D integrity levels as defined by ISO26262 specification. This introduces the requirements for safety-certified compilers and tools to be integrated as

part of the cloud-based tooling framework. Additionally, the hardware on which automotive applications are deployed is traditionally highly distributed (a modern vehicle can have 100+ ECUs and quite diverse in nature, making it difficult to achieve a decoupling of software from hardware. Finally, the existing cloud-native infrastructure, including orchestrators capable of being able to deploy the microservices to the most optimal hardware node based on enterprise-class platform awareness, needs to be extended such that they understand the capabilities of these unique embedded automotive hardware systems and can deploy the microservices in the most optimal way.

When all these requirements and issues find appropriate solutions, the developer can safely run simulations in the cloud, leveraging its inherent benefits, such as scalability and elasticity. For example, as part of the DevOps infrastructure, they would expect to be able to run a complete SiL validation by running various simulated operational design domains (ODDs) using a simulator running in the cloud, feeding a vast set of simulated data into the software under test, and rapidly verifying the output. Such a vast set of real or simulation environments could scale to verify thousands of scenarios simultaneously, launching parallel execution on thousands of cores. This is a scale that is not possible to achieve relying on embedded systems in HiL rigs.

Based on preliminary investigations with customers, we estimate that with parity in place, approximately 70 percent of the tests currently executed on HiL rigs could be moved to cloud-based SiL environments and leverage this cloud scalability.

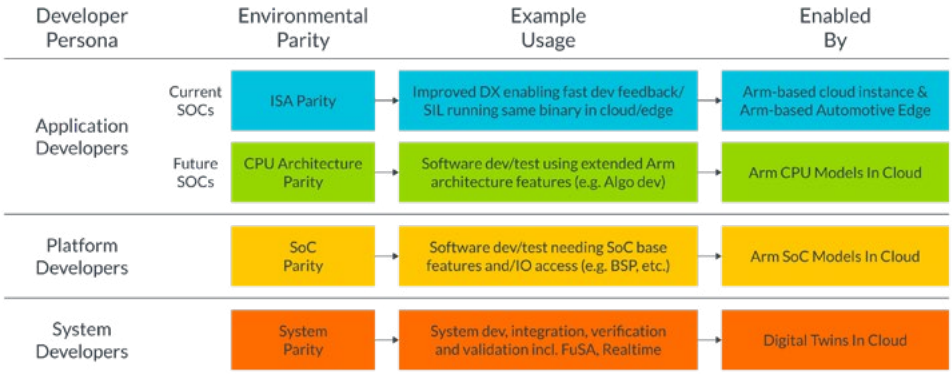
Some functionality will always need to be validated on hardware—for example, testing input/output—related to the physical nature of the embedded system, but moving from the cloud to the vehicle edge, the expectation from a system development perspective is that the same software that is validated in the cloud can be deployed through cloud-native orchestration techniques to a physical ECU to perform in-lab HiL validation before scaling it to post-sale deployment when the car is on the road.

## Cloud to vehicle edge environmental parity for enabling cloud native in automotive

One of the critical infrastructure requirements for cloud-native automotive system development is the requirement for a variety of virtual system execution environments for development, integration, and validation purposes that closely resemble the physical environment and that achieves, as much as possible, vehicle-edge environmental parity. Achieving a high level of environmental parity directly impacts the developers' feedback loop and lets them achieve a higher level of development effectiveness **[11]**. Let's dig a bit further into the environmental parity

aspects from different developer viewpoints and how they can be enabled in the cloud by reviewing different kinds of parity and personas.

Figure 3 — Cloud to vehicle edge environmental parity




## Application developer

From an automotive developer standpoint, the primary expectation from the cloud-based execution environment is to have instruction set architecture (ISA) and CPU architecture parity between the cloud and the automotive edge environment. Taking Arm-based automotive computing platforms as an example, it is now possible to host a cloud environment on the new Arm-based cloud instances provided by **AWS Graviton**. This is something of a landmark, as it is now possible to achieve a complete end-to-end Arm-based cloud-to-automotive-edge compute continuum. There are several reasons why this can be considered an ideal development environment:

- **Developer efficiencies:** Reducing developer feedback loops, as identified by developer effectiveness expert Tim Cochran **[12]**, is critical to increasing development efficiencies that impact the bottom line of companies. Some of the benefits of developing code on Arm for Arm are listed below:
  - Cross-compilation is avoided; the process of compiling code on a development system of one architecture to run on a target with a different architecture inherently adds an additional step to the develop-deploy-test process, extending the developer feedback loop.
  - Cross-compilation additionally introduces another source of potential errors to the software.
  - Hosting Arm CPU models in the cloud enables those cases where ISA parity is insufficient and CPU architecture parity is required. Emulation on a different architecture can be slow and inefficient. In an Arm-on-Arm environment, the Arm hypervisor extensions can be used directly, thus enabling faster execution of models.

- **Performance optimizations.** There is a perception that compilers take care of all the optimizations needed, but when it comes to creating truly optimized libraries or functional blocks, developers still occasionally need to make architecture-specific optimizations by hand to get the last ounce of performance for critical segments, as we see today from multiple partners and third-party ecosystem providers. In this scenario, cloud and edge parity enables a “seamless” transition of these optimizations from development to deployment.

## Platform developers

Beyond application development, another major layer of software that goes into modern vehicles is the base platform system software which includes firmware, OS/RTOS, device drivers, and related components. Platform developers need a virtual environment that simulates the SoC (system-on-chip) built using CPU cores with additional features to satisfy specific functionality) with the appropriate device interfaces along with the corresponding SoC system architecture. This can be enabled by hosting virtual SoC models in the cloud and integrated as part of the overall automotive software CI/CD framework. A key concern in this area is standardizing basic system architecture, such that off-the-shelf OS images can boot with little to no per-platform modifications. This is a challenge the Arm SystemReady certification program is directly addressing, with widespread industry support.

## System developers

As discussed earlier in this whitepaper, the workloads that run on an automotive system are mixed critical and distributed in nature. To be able to develop, verify, and validate these workloads, there must be a digital twin of the vehicle hosted in the cloud that simulates the functional and nonfunctional aspects of the system, including the real-time, safety, and performance characteristics. The digital twin must also model different operational design domains encountered in the physical environment when executed in the cloud. Major ecosystem partners that are already developing automotive digital twin technologies are making significant progress in closing the environmental parity gap.

## SOAFEE: An industry-led initiatie to drive cloud native in automotive

To make the vision of the café-based ADAS developer a reality, an ecosystem of partners across the value chain must come together and coordinate on delivering it. Arm, in concert with key partners like Amazon Web Services (AWS), recently announced the SOAFEE initiative—Scalable Open Architecture for Embedded Edge. SOAFEE offers a cloud-native architecture enhanced for mixed criticality for automotive applications and will also provide an open-source reference implementation to enable commercial and non-commercial offerings. SOAFEE is being driven by an industry-led special interest group (SIG), which is defining the SOAFEE architecture based on open standards and producing an open-source implementation. More information about this initiative can be found on the **SOAFEE website**.

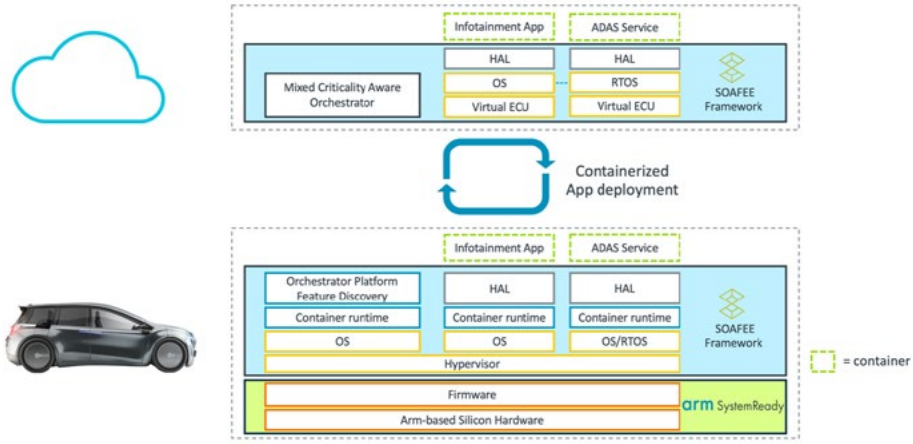
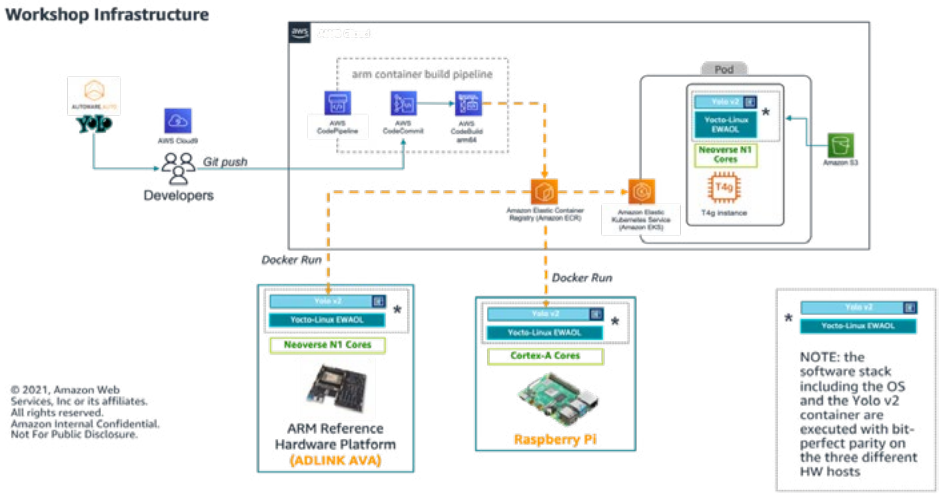



Figure 4 — SOAFEE high-level reference architecture

## Arm DevSummit workshop – demo using initial SOAFEE-based implementation

A live workshop at the Arm DevSummit 2021 **[13]** introduced the first reference implementation of SOAFEE.

Figure 5 — Automotive development pipelines with parity between cloud and vehicle edge.




As described in the architecture, this workshop introduced a novel automotive-native software development infrastructure able to execute the same containerized workload—with environmental parity—on a set of targeted compute elements: an **Amazon Elastic Compute Cloud (Amazon EC2)** instance, a **Raspberry Pi**, and an **AVA Developer Platform**.

In this architecture, we use AWS services to create a CI/CD pipeline that builds, containerizes, evaluates, and enables deployment—at scale in the cloud and on embedded devices—of a perception network, **YOLO**. This network is used as a stand-in for an automotive application workload to demonstrate the design paradigm. The specific version of YOLO used in this workshop is the YOLOv2-Tiny, running on Ubuntu Linux 20.04. Note that for the first time, the full system under test (SUT) stack included an embedded operating system (a Yocto-Linux distribution) running with native properties in the cloud, thanks to Arm's **Edge Workload Abstraction and Orchestration Layer (EWAOL)**, a reference implementation of the SOAFEE architecture.

Figure 6 — System under test: a full stack from the OS up

All the SUT components from the OS up were executed with instruction set parity, leveraging the same Aarch64 underlying architecture for all the targets. Arm's EWAOL provides users with a standards-based framework using containers for the deployment and orchestration of applications on multiple embedded platforms. This combined set of features enables:

- The cloud execution of the entire embedded software stack, from the embedded OS up, and not just of the unit of software in development (the YOLOv2-Tiny perception module itself, in this specific case).
- The SUT seamless portability from cloud to embedded edge; no more cross-compilation or emulation (and related issues, like compilation errors or performance degradation).

In fact, this approach enables the possibility to start writing and testing embedded code in the cloud, **shifting-left** the embedded development workflow, and significantly extending testing coverage by leveraging cloud scalability features (in the specific example, using AWS Batch as a way to launch multiple SUT executions in parallel).

## Conclusion and future activities

In conclusion, the software-defined vehicle is clearly no longer an over-the-horizon topic but a key current trend that is already happening, evidenced by examples across the automotive ecosystem. In the future, software is set to be the key differentiator between competing OEMs, with software developers becoming one of the most critical personas in the future of automotive system development. As the industry traverses this pivotal shift, there are challenges that can and should be solved leveraging existing cloud native technologies for automotive. With this in mind, initiatives like SOAFEE play a critical role in the adoption and acceleration of cloud native in automotive.

The first key technology enabler for cloud-native automotive software development is being able to achieve environmental parity between cloud-based development environments and the automotive edge, such that automotive software developers can develop and validate their software in the cloud and deploy it onto the edge. This will enable shift-left, scale-out, and post-sale deployment models that will enable efficiencies in automotive system development and drive innovative new business models to transform the automotive industry. Arm-based automotive target systems and AWS Graviton instances are an example of this parity, available today, and demonstrated end to end at Arm DevSummit 2021 **[14]**.

Lastly, as we march toward developing new technologies and concepts, having code to demonstrate the concepts and providing developers platforms to innovate and seed the ecosystem is crucial. In addition to the DevSummit workshop demonstrations with AWS, Arm, in collaboration with Autoware Foundation (AWF), recently announced the creation of Open AD Kit **[15]** which intends to provide a complete end-to-end development kit allowing development of autoware as a reference autonomous software stack natively on AWS Graviton instances with seamless deployment to automotive edge platforms using cloud-native orchestrators.

This is the beginning of a transformative journey for the automotive industry. Arm, AWS, and the SOAFEE SIG members will continue to collaborate on accelerating the path to cloud native in automotive. If you are equally passionate in realizing this vision, we invite you to join SOAFEE SIG. Please refer to **soafee.io/** for more details.

## References


[1] Girish Shirasat, **The Cloud-native Approach to the Software Defined Car**, Arm, September 2021

[2] Kevin Hoffman, **Beyond the Twelve-Factor App, April 2016, O'Reilly Media**, Inc.

[3] Robert N. Charette, **How Software Is Eating the Car**, June 2021

[4] Robert Day, **The Software-Defined Vehicle Needs Hardware That Goes the Distance**, June 2021

[5] Constantin Gillies, **The Big Loop: Artificial Intelligence and Machine Learning**, July 2021

[6] Alan Burns and Robert I. Davis, **Mixed Criticality Systems—A Review**, March 2019

[7] Computer Weekly—**How Containerisation Helps VW Develop Car Software**

[8] **Information is Beautiful—Million Lines of Code**

[9] Computer Weekly—**How Containerisation Helps VW Develop Car Software**

[10] **DevOps at Jaguar Land Rover**

[11] Great **Article** by Martin Fowler on Developer Effectiveness

[12] Great **Article** by Martin Fowler on Developer Effectiveness

[13] "Getting Started with Cloud-Native Automotive Software Development" workshop, recording available **here**

[14] "Getting Started with Cloud-Native Automotive Software Development" step-by-step tutorial available **here**

[15] The Autoware Foundation—**The Autoware Foundation Releases Quick-starter Kit to Accelerate Cloud-native Autonomous Development**